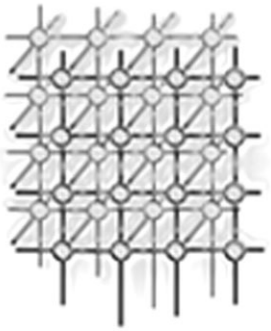
CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2006; **18**:203–213
Published online 11 October 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe.912

Coupling climate models with the Earth System Modeling Framework and the Common Component Architecture



S. J. Zhou*,†

Northrop Grumman Information Technology/TASC, 4801 Stonecroft Boulevard, Chantilly, VA 20151, U.S.A.

SUMMARY

Typical Earth system models involve coupled model components in high-performance computing (HPC) environments. In the last few years, several frameworks have been developed for HPC applications. Two of them are component-based frameworks: the Earth System Modeling Framework (ESMF) defining a component interface for Earth system models and the Common Component Architecture (CCA) defining a generic component model. The purpose of this work is to investigate the relationship between ESMF and CCA to deploy the best features of ESMF and CCA into a ESMF–CCA prototype and examine the prototype through a representative coupled climate model. Copyright © 2005 John Wiley & Sons, Ltd.

KEY WORDS: framework; climate model; model coupling

1. INTRODUCTION

A physical Earth system model typically consists of several model components, which are coupled through exchange of data. For example, the well-publicized El Niño-Southern Oscillation (ENSO) phenomenon is the outcome of atmosphere–ocean interaction. Most models solve partial differential equations on a large gridpoint set for a long time period and consequently require significant amounts of high-performance computing (HPC) resources. Since Earth system model components, such as global atmospheric circulation models, also contain many physical processes such as radiation, cloud formation, precipitation, etc., considerable time and manpower are needed to develop a production-quality model. In addition, new science such as chemistry and biology are being incorporated into

*Correspondence to: S. J. Zhou, Northrop Grumman Information Technology/TASC, 4801 Stonecroft Boulevard, Chantilly, VA 20151, U.S.A.
†E-mail: szhou@pop900.gsfc.nasa.gov

Contract/grant sponsor: NASA Earth Science Technology Office Computational Technologies Project

Received 22 June 2003
Revised 3 April 2004
Accepted 3 May 2004

the models, which requires cross-organization collaboration. Hence, models are becoming increasingly complex and difficult to update and maintain. The situation becomes even worse in the case of integrating models from different organizations. The Community Climate System Model (CCSM) development clearly shows the trend towards greater complexity, coupling of models into larger systems, and cross-organizational collaboration [1]. However, there are still many models developed and maintained by individual organizations [2]. A software framework to facilitate coupling models and to make modes interoperable would be a great benefit to these and similar activities.

In the past few years, scientists and researchers realized that many HPC applications can be divided into two parts: one is computer science and the other is physical science. Computer science addresses the issues of computer architecture, parallel communication, I/O, error checking, visualization, etc. Physical science covers the issues of solvers for various partial differential equations (PDEs), FFT routines, numerical grids, etc. Following this philosophy, Parallel Object-Oriented Methods and Applications (POOMA) was developed at the Los Alamos National Laboratory (LANL) to provide an object-oriented framework for applications in computational science requiring high-performance parallel computers [2,3]. Similarly, Overture framework was developed at the Lawrence Livermore National Laboratory (LLNL) and LANL. The project is now consolidated into the LLNL [2,4]. Overture is an object-oriented software system for solving PDEs in a serial and parallel environment. In the astrophysics community, the Cactus framework was also developed [2,5]. The goal of the Cactus framework is also to enable an application to run in a serial and parallel environment and across different computer architectures. There are also domain-specific frameworks and software packages, for example, Distributed Data Broker [2], Flux Coupler [2], Goddard Earth Model System [2], and Flexible Modeling System [2,6] for Earth system models. POOMA, Overture, and Cactus have proved that a problem-solving environment for parallel computing can be developed and can be useful. The Distributed Data Broker, Flux Coupler, Goddard Earth Model System, and Flexible Modeling System show frameworks and software packages can facilitate Earth system modeling in the aspects of development and coupling. However, there is a lack of community-level standard for defining Earth system models, especially interfaces for model interaction, and an infrastructure for supporting model development and coupling. Without such a standard and infrastructure, coupling models or interoperating models across organizations is very time-consuming and difficult.

To achieve that goal, NASA's Earth Science Technology Office/Computational Technology project has funded the development of the Earth System Modeling Framework (ESMF). The ESMF project enables close collaboration from major U.S. Earth system modeling organizations: the NSF/National Center for Atmospheric Research (NCAR), NASA/Goddard Space Flight Center, the Massachusetts Institute of Technology, the University of Michigan, DOE/Argonne National Laboratory, DOE/LANL, the NOAA/Geophysical Fluid Dynamics Laboratory, and the NOAA/National Centers for Environmental Prediction. The ESMF software will consist of a superstructure for coupling and exchanging data between component models (e.g. atmosphere, ocean) and model subcomponents (e.g. physics, dynamics); and an infrastructure consisting of (1) data structures for representing grids and fields and (2) an optimized, portable set of low-level utilities. The data constructs and low-level utilities will be used by the coupling superstructure and may also be used separately to compose scientific applications. Conceptually, an application running under ESMF may be thought of as a sandwich, with the upper and lower layers provided by the ESMF and the middle layer provided by the application developer. The ESMF superstructure sits above the components of an application, controlling inter-component data transfer and sequencing. The ESMF infrastructure lies

below the components, offering integrated tools for intra-component communication, error handling, time management, profiling, and other standard modeling functions. More information on ESMF can be found in [7].

To encourage the collaboration among its laboratories, DOE also funded the Center for Component Technology for Terascale Simulation Software (CCTTSS) as an Integrated Software Infrastructure Center (ISIC) under the Scientific Discovery through Advanced Computing (SciDAC) program. The members are from LLNL, LANL, Argonne, Oak Ridge, Pacific Northwest, and Sandia National Laboratories, Indiana University and the University of Utah. CCTTSS is dedicated to the development of a component-based software development model suitable for the needs of high-performance scientific simulation, particularly the Common Component Architecture (CCA). The CCA Forum was organized to define a minimal set of standard interfaces that a high-performance component framework has to provide to components, and can expect from them, in order to allow disparate components to be integrated to build a running application. Such a standard will promote interoperability between components developed by different teams across different institutions. More information on CCA can be found in [8].

These two emerging component-based frameworks, ESMF and CCA, have a common goal: to promote interoperability between components developed by different organizations. However, CCA provides a more generic component interface that is not specific to any one type of application, while ESMF is designed with specific component interface methods that are common to Earth system model components. To investigate how an Earth system model component that is ESMF compliant can be supported in CCA, we have designed and developed an ESMF–CCA Prototype. A representative coupled climate model was developed and is used to explore the compatibility issues between ESMF and CCA. In this paper, we will describe our ESMF–CCA Prototype and present our findings.

2. BRIEF REVIEW OF FRAMEWORK ARCHITECTURE

POOMA, Overture, Cactus, and ESMF use a layered structure to isolate functionality, although implementation varies due to the programming languages they use. Basically functionalities such as I/O, parallel communication, computer architecture are at the lower level, while PDE solvers and other numerical algorithms are at the high level. To encapsulate parallel programming, POOMA, Overture, and Cactus specially design an array to contain domain decomposition information. That is, a user does not need to know in which processors the data are located (ideally). The drawback is that a user has to use such an array in order to access high-level features such as PDE solvers provided by the frameworks. That impedes the usage of the frameworks since a production-level code may not easily accommodate such a requirement due to the issues of programming languages and internal code structures. A user may view this requirement as ‘intrusive’. CCA acknowledges such an issue. Its architecture attempts to minimize ‘intrusiveness’. The ‘Provide–Use’ system design pattern is used and a registering service is provided. In the period of compilation, a user informs the CCA framework what he or she needs and a provider tells the CCA framework what he or she can provide. During the runtime, a user has a choice to dynamically pick the functionality. In this way, a user determines what he or she needs or provides. We will further discuss CCA and ESMF in the following sections.

3. DESCRIPTION OF ESMF–CCA PROTOTYPE

Coupling model components in an extensible and flexible way is a very challenging and domain-specific problem. CCA components interact by adhering to the 'Provide–Use' design pattern. This means that each component publishes the functionality that it allows other components to access. These published methods are known as Provides-Ports. Each component also publishes the functionality that it needs to have other components perform for it. These published methods are known as Uses-Ports. Conceptually a 'port' can be thought of as a contract between components of a system. It is equivalent to Java interfaces and pure abstract class definitions in C++. The CCA framework includes one additional type of port, a 'go' port, the starting point for executing systems of components. A driver component implements a 'go' port and schedules and controls the running sequence of components. A CCA-compliant framework, like CCaffeine, is responsible for connecting and managing ports.

To be a CCA component, the component has to override one pure virtual function, setService, of a CCA framework. That action essentially registers the component to the CCA framework. After that, the component can inform the framework what ports it provides and what ports it uses. Through the CCA framework, any component can know what others can provide and subsequently gets the needed ports and uses its functions.

A typical Earth system model such as an atmospheric model has three major functions that it must perform: initialization, run, and finalization. An initialization routine provides the functionality to prepare a model for simulation and includes initialization of parameters and boundary conditions. A run routine provides the functionality to allow the model to perform its time evolution. A finalization routine provides the functionality to safely shut down operations, clean up memory, and close any open files. ESMF's component model requires an ESMF-compliant model component to provide these functions, and the ESMF superstructure is responsible for mapping these routines to the standard interfaces that other ESMF components expect to call. In addition, ESMF provides a standardized, self-describing format for data exchange among model components through the ESMF_State data type.

A key difference between the CCA component environment and the ESMF component environment is ESMF's requirement for a user to supply additional standard methods (e.g. initialization, run and finalization) beyond those required for registering the component in the framework. ESMF provides a standard component adapter class, ESMF_Comp, that maps the standard component interface to the functionality implemented by the model developer's code. CCA is not application specific and does not require any additional methods or standard interfaces.

Our ESMF–CCA Prototype will utilize CCA's 'Provide–Use' design pattern to create and couple CCA components that use the ESMF standard component interfaces described above. That is, Initialize(), Run(), Finalize(). In addition, data to be exchanged between components have a data type similar to ESMF_State. This will allow ESMF-compliant components to run within a CCA component framework. CCA's Uses–Provides design pattern has been successfully used for coupling various components in a non-intrusive way [8].

4. DESCRIPTION OF A REPRESENTATIVE COUPLED CLIMATE MODEL

To explore the compatibility issues between ESMF and CCA and test the ESMF-CCA Prototype, we have designed and developed a 2D Representative Coupled Atmosphere–Ocean Model (RCAOM).

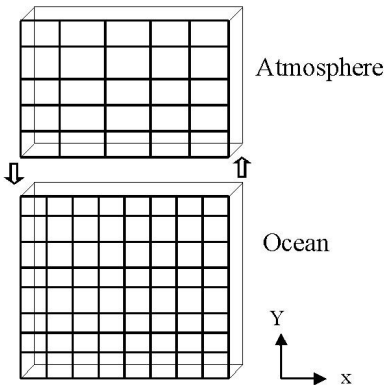


Figure 1. Illustration of the geometry of a representative 2D coupled climate model.

RCAOM is computationally similar to, but physically simpler than, the Shallow Water Model [9]. The design philosophy is to keep the essential features of coupled climate models (e.g. different grid resolution, timestep, and iteration number) and to simplify the features of physics and numerical algorithms as much as possible. The 'Atmosphere' model component uses a coarse rectangular grid, while the 'Ocean' model component uses a fine grid (see Figure 1). Each model component consists of a forward finite-difference advection scheme. Periodic boundary conditions along the x direction are used in each model component. The upper boundary of the 'Atmosphere' model component and the lower boundary of the 'Ocean' model component are free. Data are exchanged at the overlapped boundary along the y direction. In addition, each model component can choose an independent timestep. To couple an atmosphere with an ocean model component, we also developed two simple couplers: one from atmosphere to ocean and another from ocean to atmosphere. Basically, the coupler transforms data on the overlapped boundary along the y direction from one grid to another grid. Each component is represented in a C++ class since CCA's Ccaffeine framework supports C++ naturally. We will use ESMF utilities to address the issue of language interoperability between C++ and Fortran in a follow-on project.

5. RESULTS

A typical sequential coupling sequence between an atmosphere model and an ocean model is depicted in Figure 2, where t_{global} is the time advance in one coupling cycle and n_{cycle} is the number of coupled cycles. At the beginning of a simulation, each component is created. In the ESMF–CCA Prototype, we use CCA's setService utility to register the components of atmosphere, ocean, coupler from atmosphere to ocean (CplAtmXOcn), and coupler from ocean to atmosphere (CplOcnXAtm). In addition, data exchange is implemented through import or export states that are of the data type similar to ESMF_State (see Figure 3). The atmosphere component first provides its data at its boundary,

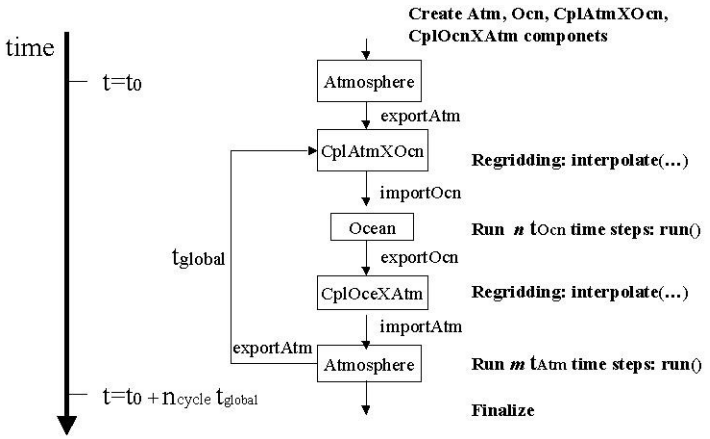


Figure 2. A typical sequential coupling between an atmosphere and an ocean model component.

exportAtm, to the coupler, CplAtmXOcn. CpleAtmXOcn uses exportAtm, transforms it into importOcn with an interpolation routine, and provides importOcn to the ocean component. With importOcn, the ocean component runs its finite-difference advection scheme for n*t_{Ocn} timesteps while satisfying its boundary conditions in the x and y directions. After that, the ocean component provides its data at the boundary, exportOcn, to the coupler, CplOcnXAtm. The coupler, CpleOcnXAtm uses exportOcn, transforms it into importAtm, and provides importAtm to the atmosphere component. Similar to the ocean component, the atmosphere component uses importAtm, runs its finite-difference advection scheme for m*t_{Atm} timesteps while satisfying its boundary conditions in the x and y directions. Then, the atmosphere component provides its data at the boundary, exportAtm, to the coupler, CpleAtmXOcn. This completes a coupling cycle with the time advanced, $t_{\mathrm{global}}(= \mathrm{n}*t_{\mathrm{Ocn}} + \mathrm{m}*t_{\mathrm{Atm}})$.

To test the ESMF–CCA Prototype, the 2D representative coupled climate model is integrated into the ESMF–CCA Prototype. The first step to integrate the model is to decompose the original application into components and sort out the relationships among model and coupler components. Componentization appears to work quite naturally with the climate model since the ocean model and atmosphere model have their own physical processes. Once identified and componentized, integrating those components into the ESMF–CCA Prototype is straightforward. Careful design of each component should be able to modularize and encapsulate each model well. Well-defined interfaces enable the model to communicate with other components in a flexible way. One of the examples is to dynamically assemble model components without modifying their codes (i.e. 'plug-and-play').

To illustrate a simulation running in the environment of the ESMF–CCA Prototype, we choose the simulation configuration as follows: the grid of the Atmosphere component is the same as that of the Ocean component. The field variable such as wind speed is defined on the grid point. The grid sizes for atmosphere and ocean components are 16×16 and 31×31, respectively. For simplicity, these two grids also align with each other on their overlapped boundary. The initial condition for the Atmosphere

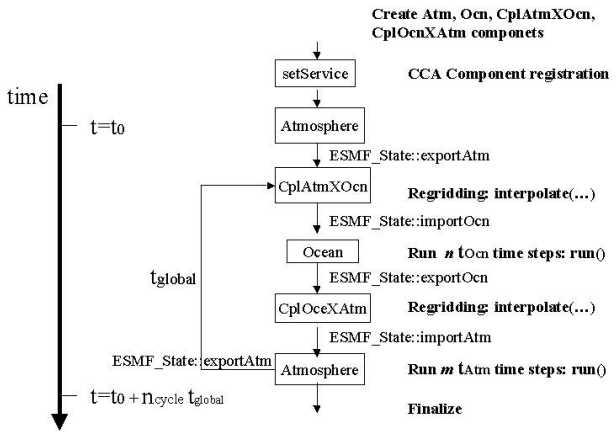


Figure 3. The flow diagram shown in Figure 2 implemented in the ESMF–CCA Prototype.

component is zero everywhere while a Gaussian distribution with its peak at the center of the grid is used for the Ocean component. In each coupling cycle, the Atmosphere component iterates five times before providing its data at the boundary to the coupler while the Ocean component iterates once. This scheduling configuration is to imitate real-world simulations of the coupling between atmosphere and ocean models.

In the ESMF–CCA Prototype, each component is coded, compiled as a shared library (.so), and stored in an individual directory. With a script, a user can list those components to be selected from. After running the script, those components are loaded in and displayed in the CCA GUI environment as seen on the left side of Figure 4 (palette). The user can select a component from those that have been loaded and drag it to the right side (arena). Connections can then be made between various components in the arena by clicking on a component's Provides-Port and then clicking on another component complimentary Uses-Port. The red wires show the Provides–Uses relationship of the components and do not necessarily indicate the actual sequence of data flow between the components. The driver implementing the 'go' port controls the sequence of execution cycle and data flow in this simulation.

During the simulation, the atmosphere and ocean components output data after each coupling. Currently, we process simulation data after the simulation finishes. In a follow-on project, we will develop a component that can be used to visualize the simulation data in real time. Figures 5 and 6 clearly show that a 'wave' in the ocean model component has been advected into the atmosphere model component (which had a zero field at the beginning) after 10 coupling cycles.

To demonstrate the flexibility of the ESMF–CCA Prototype, we have also developed several ocean model components with the same interface but with a different implementation of the initial condition (e.g. the amplitude and peak location of the Gaussian distribution). With the ESMF–CCA Prototype, a user can dynamically select any one of several ocean model components to participate in the simulation (see Figure 7). That is, the exact feature required by climate modeling since climate prediction is

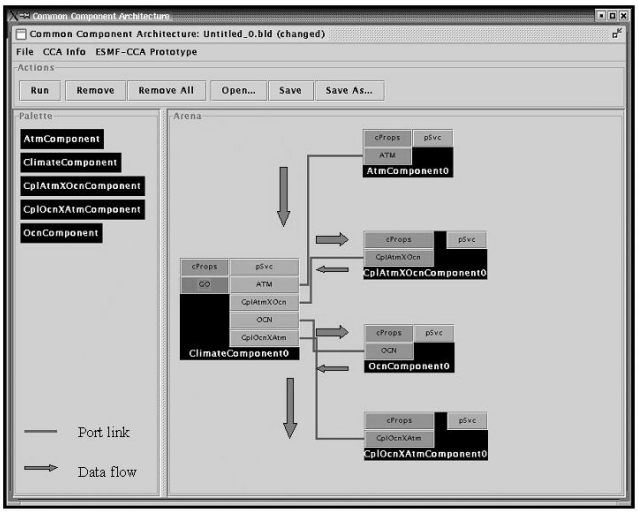


Figure 4. Component relationship via a CCA wiring diagram in a simulation of coupled atmosphere and ocean model components.

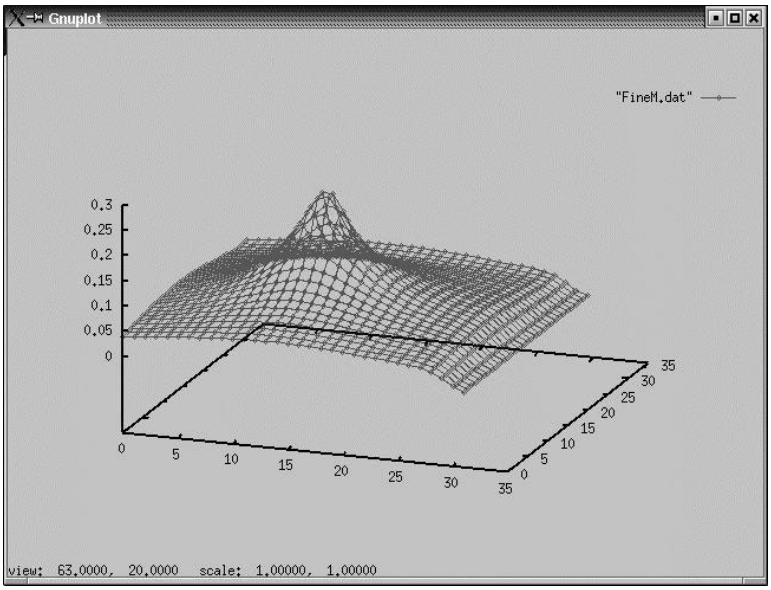


Figure 5. Simulation results of the ocean component after 10 coupling cycles. The x and y axes represent the grid location, and the z axis is one physical variable such as wind speed.

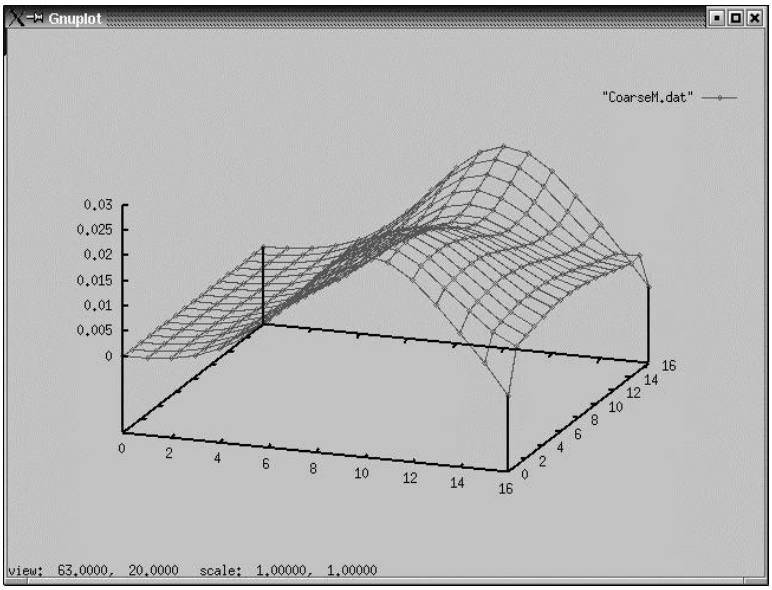


Figure 6. Simulation results of the atmosphere component after 10 coupling cycles. The x and y axes represent the grid location, and the z axis is one physical variable such as current speed.

based on an ensemble of simulations with different initial conditions and parameters. The ESMF–CCA Prototype can also support this kind of operation: once a simulation is completed, the components in Arena can be removed, rearranged, and rewired to perform another simulation. In short, the well-defined model component interface of ESMF–CCA Prototype facilitates coupling of various models.

6. DISCUSSION

As a result of integrating a representative coupled climate model into the ESMF–CCA Prototype mentioned above, it appears that coupled climate models can be decomposed into modular components and can be naturally supported in component-based frameworks such as ESMF and CCA. ESMF and CCA complement each other nicely. ESMF standardizes the interfaces to the Earth system model components and provides utilities and standard data formats for building and coupling these components. CCA provides a flexible method for controlling and managing components and their coupling. The ESMF's component design is important for creating an environment where similar components can be exchanged easily. This will allow comparison of components developed by various organizations without major changes to the climate model. However, the ESMF implementation does not provide the complete flexibility of easily connecting any component's Provides-Port to any other components matching Uses-Port like CCA. CCA components can also be extremely small. All ESMF components must implement a non-trivial set of standard methods to be ESMF compliant.

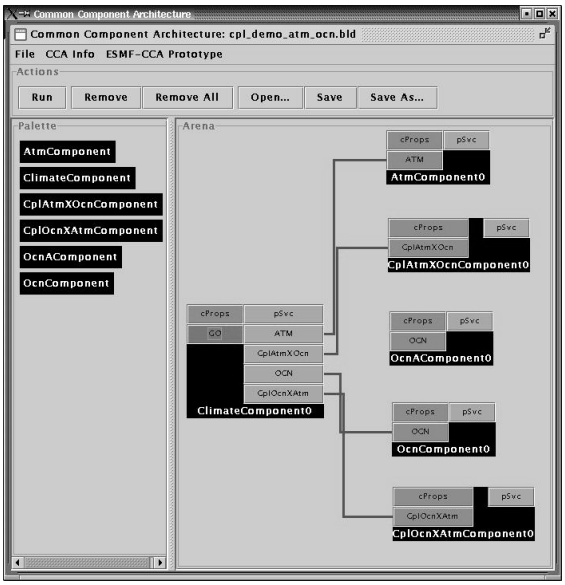


Figure 7. Ocean model components are available to be selected during the running time of a simulation.

Once ESMF compliant Earth system model components are available from the ESMF application development teams, we will integrate one or more of them into the CCA framework.

Currently we are replacing our representative coupled atmosphere–ocean model with an intermediate climate model, the Cane–Zebiak model, which is well known for its prediction of the El Niño phenomenon [10]. The existing Cane–Zebiak model was written in Fortran 77. The data exchange is through F77's common block (i.e. global variables), which is the most difficult problem for componentization. Solving this problem requires a comprehensive understanding of the model. This is a common, important issue that a framework user has to face in the process of integrating a model into frameworks. We are using the F77 subroutine argument list to exchange the data between atmospheric and oceanic model components. Even though the process is slow, tedious, and painful, it is necessary for the Cane–Zebiak model to be extensible and flexible so as to interact with other models.

7. CONCLUSION

We have successfully developed a prototype with the features of ESMF and CCA and demonstrated that climate models can be naturally supported in a component-based software framework through integrating a representative coupled climate model into the ESMF–CCA Prototype. We found that CCA and ESMF are compatible and can benefit each other.

ACKNOWLEDGEMENTS

This is project is supported by the NASA Earth Science Technology Office Computational Technologies Project. We would like to thank A. DaSilva, B. Womack, and G. Higgins for helpful discussion in the development of ESMF-CCA Prototype and W. Elwasif and D. Frantz for installing CCA's Ccaffeine framework and R. Armstrong, B. Allan, and J. Ray for technical support in using the Ccaffeine framework. We would also like to thank the Sandia National Laboratory for hosting 2003 CCA Climate Week where the architecture difference between CCA and ESMF was further investigated.

REFERENCES

1. Community Climate System Model. http://www.ccsm.ucar.edu/ [4 February 2004].
2. Survey and Analysis of Frameworks. http://ct.gsfc.nasa.gov/esmf_tasc/index.html [4 February 2004].
3. Parallel Object-Oriented Methods and Applications. http://www.codesourcery.com/pooma/pooma [4 February 2004].
4. Overture Framework. http://www.llnl.gov/CASC/Overture [4 February 2004].
5. Cactus Framework. http://www.cactuscode.org [4 February 2004].
6. Flexible Modeling System. http://www.gfdl.gov/∼fms [4 February 2004].
7. Earth System Modeling Framework. http://www.esmf.ucar.edu/ [4 February 2004].
8. Common Component Architecture. http://www.cca-forum.org/ [4 February 2004].
9. Matsuno T. Quasi-geostrophic motions in the Equatorial area. *Journal of the Meteorological Society of Japan* 1966; **44**(1):25–42.
10. Zebiak S, Cane M. A model El Niño-Southern Oscillation. *Monthly Weather Review* 1987; **115**(10):2262–2278.

